Занятие 5

Разработка тестов

Какие бывают тесты

Перед вами обыкновенная ручка.
Давайте подумаем, как её можно протестировать?

Тестирование ручки

1. Тесты на основе требований (requirements based tests)

Требование	Тест
Пользователь может поменять стержень	Извлекается и вставляется ли в ручку стержень?
Ручку можно зацепить за карман	Присутствует ли держатель, позволяющий цеплять ручку за край кармана?
Ручка с кнопкой и возможностью спрятать стержень в корпус	Переключается ли ручка из рабочего в нерабочее положение?

Тестирование ручки

2. Функциональные тесты (functional test)

    1 Вставить в ручку стержень

    2 Переключить в рабочее положение

    3 Написать несколько слов

    4 Переключить в нерабочее положение

2 Взять ручку в руки

3 Раскрутить корпус

4 Извлечь стержень

Тестирование ручки

3. Сценарные тесты (scenario tests). Как ручку может использовать:
   - Секретарь
   - Преподаватель
   - Студент
   - Школьник
   - Прораб
   - Сантехник
   - Милиционер
   - Моряк …

Тестирование ручки

4. Негативные тесты (negative testing)

- Что произойдёт, если препятствовать выходу стержня в рабочее положение?
- Какое усилие и где надо приложить к ручке, чтобы её сломать?
- Если стержень застрял, легко ли его извлечь?
- Что произойдёт, если писать по стеклу, асфальту?

Тестирование ручки

5. Тесты интерфейса (interface tests, GUI tests)
    - Измерения: высота, ширина, длина, вес
    - Цвет
    - Читаемость логотипа фирмы-производителя
    - Материал

Тестирование ручки

6. Тесты удобства использования (usability tests)

- Как быстро пользователь понимает, как пользоваться ручкой?
- Как быстро пользователь привыкает к этой ручке?
- Легко ли понять, какие стержни подходят к ручке?
- Легко ли заменить стержень?
- Может ли ручкой пользоваться левша?
- Не смазываются ли чернила, если пишет левша?

Тестирование ручки

7. Тесты документации (packaging/documentation tests)

- Ясно ли видно на упаковке, что внутри?
- Легко ли открыть упаковку?
- Есть ли какие-то особые требования к упаковке?
- На сайте, в каталоге, на упаковке написано, нарисовано одно и то же?
- Текст на упаковке и в гарантийном обязательстве – на одном и том же языке?
- На упаковке и в документации нет грамматических ошибок, опечаток и т.д.?

Тестирование ручки

8. Стресс тесты (stress tests)

- При какой температуре расплавится пластиковая часть ручки?
- При какой температуре потечёт стержень?
- При какой температуре ручка перестаёт писать?
- Какое воздействие нужно применить к ручке, чтобы сломать её?
- Пишет ли ручка под водой? А по мокрой бумаге?
- Если ручку уронить в песок – что произойдёт?
- А если уронить со стола?
- А если из окна офиса?

Тестирование ручки

9. Тесты производительности (performance tests)

   - Сколько текста можно написать ручкой в единицу времени?
   - Как быстро ручку можно привести в рабочее положение?
   - Как много раз ручку можно переключить из нерабочего в рабочее положение, прежде чем её начнёт заедать?

Тестирование ручки

10. Конфигурационные тесты (configuration tests)

    - Какие стержни подходят к нашей ручке?
    - На каких поверхностях она может писать?

Чек-лист

Чек-лист (checklist) - список проверок без описания шагов

Упрощенная форма тест-кейса

- у него нету четкой структуры, вернее их очень много
- из важных характеристик – краткость и понятность (простота)
- быстрота создания и понимания

Пример чек-листа

Операции с файлами	ok	
Создание файла	ok	
Открытие файла	ok	
Сохранение документа	ok	
Печать	ok	
Редактирование файлов	bugs	
Отмена	ok	
Копирование	ok	
Вырезание	ok	
Вставка	ok	
Удаление	ok	
Поиск	fail	bug #123
Поиск с заменой	fail	bug #126
Вставка даты	ok	
Форматирование	ok	
Перенос строки	ok	
Изменение шрифта	ok	

Определения тест-кейсов

Тест кейс – (тестовый случай) совокупность шагов, условий и параметров созданных для проверки работоспособности функции или ее части

A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement

(Набор тестовых входных данных, условий выполнения и ожидаемых результатов, разработанных с конкретной целью, такой как проверка некоторого пути выполнения программы или проверка соответствия некоторому требованию)

Далее «тест-кейс» = «тест»

Структура тест-кейса

1. Номер (number) или идентификатор (id)
2. Связанное с тестом требование (related requirement)
3. Модуль (Feature)
4. Имя (name)
5. Предусловия (Preconditions)
6. Шаги (Steps)
7. Ожидаемый результат (Expected result)
8. Статус (Passed, Failed, Blocked)
9. Приоритет (Priority: smoke, critical…)
10. Связанный с тестом баг (если есть) (related bug)
11. Постусловия (Postconditions)

Пример тест-кейса

#	Module	Test Case Name	Preconditions	Steps	Expected Result	Status	Comments
118	Каталог товаров	Изменить количество товаров на странице	Зайти на сайтdemo.shopto.by	1. Открыть каталог товаров. 2. В поле **Товаров на странице** выбрать разные значения.	Количество товаров на странице меняется	Passed	

#	Name	Preconditions	Steps to Perform	Expected Result	Priority	Win 7, IE 8	Win 7, IE 9	Win 7, IE 10	Win 7, FF
Доступ и GUI формы									
1	Доступность формы	Есть доступ в интернет	1. Открыть главную страницу Яндекс: http://yandex.by 2) Ввести любой запрос или оставить поисковую строку пустой и нажать «Найти» 3) Под строкой поиска перейти по ссылке «Расширенный поиск» http://yandex.by/search/ advanced?numdoc=10&text=test&lr=157	Форма РП открыта: Вид формы.jpg	High				Hold
2	Все заявленные элементы есть на форме	Форма РП доступна	1. Открыть форму РП Яндекс 2. Проверить наличе на форме заявленных элементов: - кнопок, - полей ввода (input fields) - ссылок - чекбоксов - примеров \ подказок	На форме есть следующие элементы: Вид формы-поля.jpg					

Документирование тестов

Результатом документирования тестов является **тест-кейс**

Набор тест-кейсов – **Test Suite**

Test Suite объединяет тесты по какому-то принципу:

- Модулю
- Функционалу
- Виду тестирования (функциональные, для регресса …)
- Приоритету …

На основе чего писать тест-кейсы?

1. Требования
2. Здравый смысл
3. Опыт
4. Работающее приложение (приложение-прототип, прототип)

Свойства тест-кейсов

Тест-кейсы могут быть:

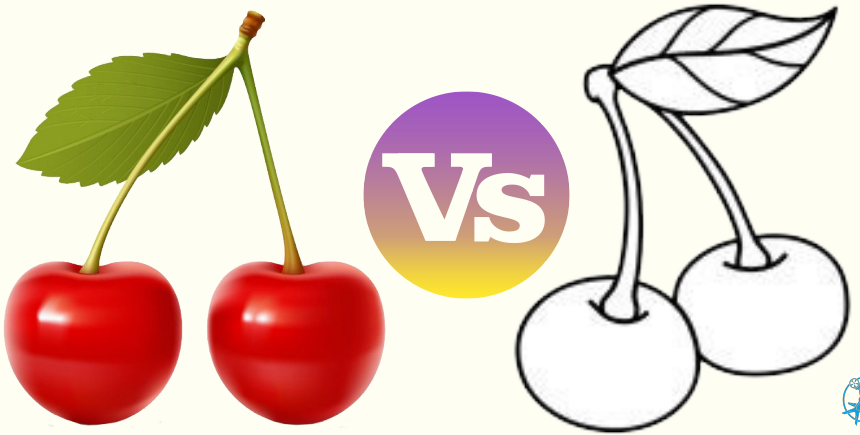
1. Специфичными или общими (степень детальности)
2. Простыми или сложными
3. Независимыми или связанными друг с другом
4. Позитивными или негативными

Зачем нужны тест-кейсы

1. Тест-кейсы – хороший способ хранения проектной информации
2. Написание тест-кейсов – один из способов протестировать проектную документацию ещё до выхода первого билда
3. Наличие тест-кейсов ускоряет регрессионное тестирование
4. Тест-кейсы – прекрасный способ быстро ввести в курс дела новичка или сотрудника, только что подключившегося к проекту
5. Имея тест-кейсы, мы можем в любой момент «вспомнить», что мы делали месяц, полгода, год назад.
6. Тест-кейсы позволяют легко отслеживать прогресс (X% тестов выполнено, Y% тестов прошло (завалилось), Z% требований покрыто тестами)

Детальность

- Когда все детали прописаны до мелочей, тест легко воспроизводить
- Когда тест прописан очень детально снижается вероятность обнаружить ошибку (меньше будем исследовать)
- Слишком общий тест-кейс сложно выполнять (особенно новичку)



Детальность

- Если в тесте прописано много мелких деталей, возрастает время его создания и поддержки
- Однако недостаток деталей может усложнить работу новичка
- Интеграционные тесты обычно более общие

Простота или сложность

Рассмотрим на примере.
Где в ниже перечисленном простые тест-кейсы, а где - сложные?

Тест 1:

1. Откройте файл «1.txt». Файл открыт
2. Введите слово «Дом». Появляется слово «Дом»
3. Сохраните файл

Тест 2:

1. В документе размером более 100 Мб создайте таблицу 100×100
2. В ячейку 50×50 вставьте картинку размером 30 Мб,

Простота или сложность

Каковы преимущества простых тест-кейсов?

- Их легко выполнять
- Они понятны новичкам
- Они упрощают диагностику ошибки
- Они делают наличие ошибки очевидным

Простота или сложность

Каковы преимущества сложных тест-кейсов?

- Больше шансов что-то сломать
- Пользователи, как правило, используют сложные сценарии
- Программисты сами редко проверяют такие варианты
- Следует постепенно повышать сложность тестов

Независимость или связанность

Каковы преимущества независимого самостоятельного тест-кейса?

- Его легко и просто выполнить
- Такие тесты можно выполнять даже после краха других тестов
- Такие тесты можно группировать любым образом и выполнять в любом порядке

Каковы преимущества наборов тесно связанных тестов?

- Они имитируют работу реальных пользователей
- Они удобны для разбиения на части тестов с большим количеством шагов
- Следующий в наборе тест использует данные и состояние

Хороший тест

Принято считать **хорошим** тестом - **независимый тест**

Позитивность или негативность

Позитивные тесты проверяют, что приложение делает ТО, на ЧТО оно РАСЧИТАНО (т.е. такие тесты используют корректные данные и условия выполнения).

Нацелены подтвердить

Негативные тесты проверяют работу приложения в нестандартных условиях (с некорректными данными или командами или при работе в некорректных условиях).

Нацелены сломать

Язык написания тест-кейсов

- Используйте активный залог
  (open, paste, click)
- В русском языке используйте
  безличную форму:
  открыть, нажать, загрузить
  (вместо откройте, нажмем, загружаю
- Пишите короткими пунктами
  (одно действие = один шаг)

- Используйте простой технический стиль
- Обязательно указывайте точные названия элементов

Рекомендации

- Начинайте с простых очевидных тестов
- Затем переходите к более сложным тестам
- Выделяйте классы эквивалентности
- Помните о граничных условиях
- Организуйте сценарий логично и последовательно
- Используйте один тест для ОДНОЙ проверки
- Используйте средства форматирования (выделяйте, добавляйте картинки)

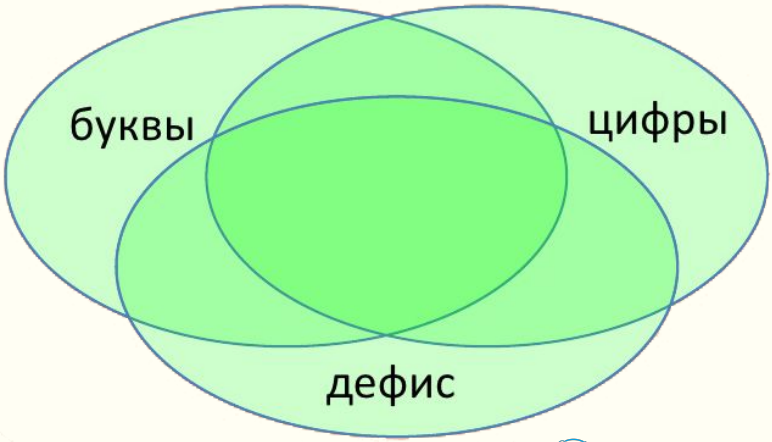
Критерии хорошего тест-кейса

- Не выполняет ненужных действий
- Не является избыточным по отношению к другим тестам
- Исследует соответствующую (ту, которую надо) область приложения
- Позволяет легко диагностировать ошибку
- Независим (каждый тест-кейс – это индивидуальный сценарий с точкой входа и точкой выхода)

Классы эквивалентности

Класс эквивалентности
(equivalence class) - набор тестов, со схожими входными данными, шагами воспроизведения и одним ожидаемым результатом

Класс эквивалентности – множество, все элементы которого программа обрабатывает одинаково



Признаки эквивалентности

Несколько тестов эквивалентны, если:

- Они направлены на поиск одной и той же ошибки
- Если один из тестов обнаруживает ошибку, другие её тоже, скорее всего, обнаружат
  (и наоборот, не обнаружит один, не обнаружат все)
- Тесты используют одни и те же наборы входных данных
- Для выполнения мы совершаем одни и те же действия
- Тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние
  (например, открытие валидных файлов одного типа и схожего объема, но с разным содержанием)
- Все тесты приводят к срабатыванию одного и того же блока обработки ошибок («error handling block»)

Граничные условия

Граничные условия (границы) - это те места, в которых один класс эквивалентности переходит в другой

Граничные условия очень важны, т.к. именно в этом месте чаще всего и будут ошибки

Эквивалентность и границы

Проверить, как работает поле, в которое можно ввести целое число от 1 до 99

Классы эквивалентности здесь:

- Любое целое число начиная с 2 и по 98.
  Как правило, это будет середина числового отрезка
  (Позитивный тест)
- Граничные значения 1 и 99 (Позитивный тест)
- Любое число меньше 1 (Негативный тест)
- Любое число больше 99 (Негативный тест)
- Другой класс - буква (Негативный тест)
- Другой класс – спецсимвол ~`!”@’#$;%:^&?*()[]{},.\/+=-_
  (Негативный тест)

Пример для обсуждения

Какие тесты нужно провести?

Пример для обсуждения

- Файл фото формата (jpg. png…) и НЕ фото формата (txt., mov.,)
- Граничный формат - pdf
- Пустой файл / не пустой файл
- Уже открытый файл (использует др. приложение) и файл «в покое»
- Поврежденный файл и не поврежденный
- Файл неверного формата (По расширению и/или реальному содержимому)

Тестовые набор [Test Suite]

Тестовый набор (Test Suite) – a set of related tests, usually pertaining to a group of features or software component (module) (набор связанных тестов, как правило, относящихся к группе функций или программного компонента)

Хороший тестовый набор всегда следует некоторой логике, например: типичному использованию приложения, удобству тестирования, распределению функций по модулям…

Рекомендации

- Пишите набор для отдельной части приложения (модуля)
- Помните, что заголовки тестов отражают их суть. Правильно формулируйте и оформляйте заголовки
- Помните о необходимых приготовлениях к тесту. Описывайте их (preconditions)
- Не повторяйте в нескольких тестах одни и те же шаги, выносите их в preconditions
- Четко формулируйте Expected result

Когда пишете тесты

- Copy-paste
- Если по ходу разработки тестов возникают вопросы, пишите их прямо в документ с тестами, помечая цветом
- Используйте «косметику» (жирный, подчёркнутый, наклонный шрифт, разные цвета)
  Это повышает читаемость документа
- По-максимуму используйте возможности ПО, в котором вы разрабатываете тесты (группировки, фильтры, ссылки)
- Обязательно прописывайте в файле историю изменения, исправления выделяйте цветом

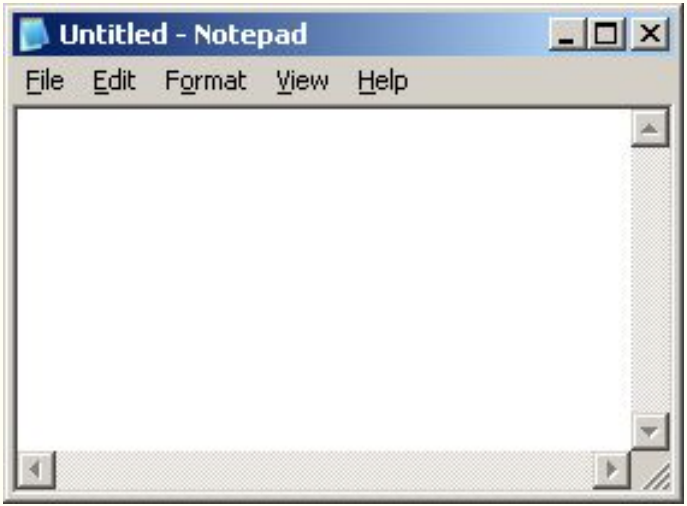
Как прийти к тест-кейсу

- Сбор информации (требования, мок-апы…)
- Делим приложение на модули, выделяем функции
- Пишем чек-листы
- Детализируем чек-листы в тест-кейсы

Работа с тест-кейсами

- Начинайте как можно раньше, ещё до выхода первого билда
- Разбивайте приложение на отдельные части/модули
- Для каждой области/модуля пишите чек-лист
- Пишите вопросы, уточняйте детали, добавляйте «косметику» используйте copy-paste
- Обновляйте тесты, как только обнаружили ошибку или изменилась функциональность

Первый тест-кейс. Notepad

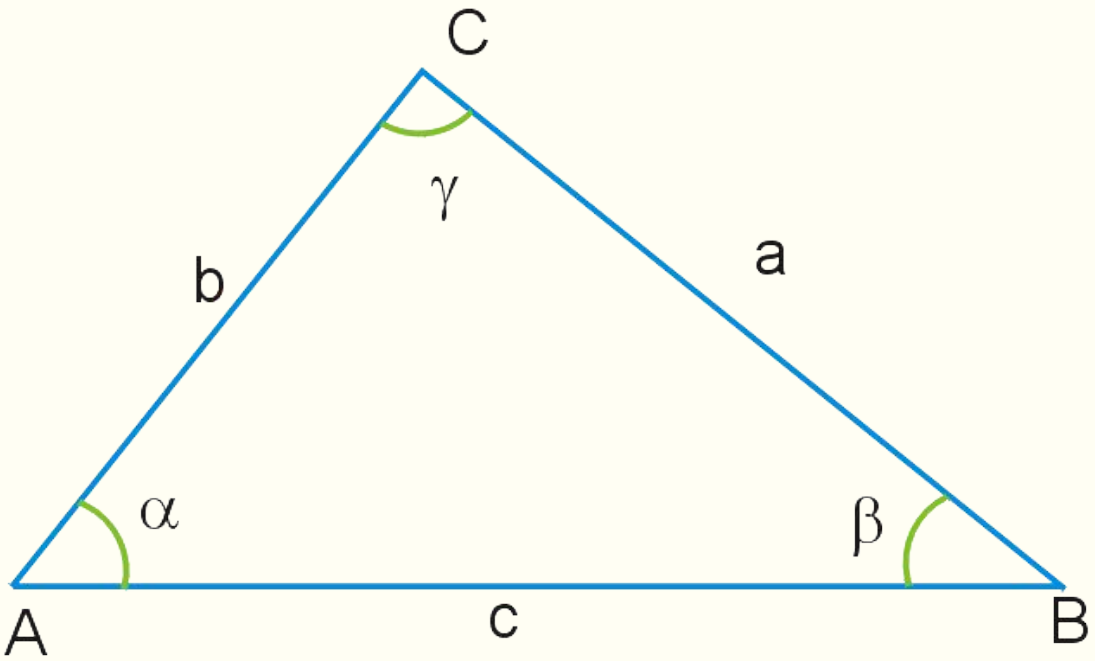


	A
1	Check
2	Запустить приложение
3	Создать новый файл
4	Ввести текст
5	Сохранить файл
6	Распечатать файл
7	

Пункты грамотно сформированного чек-листа - готовые заголовки тест-кейсов

iD	REQ	Название	Предусловия	Шаги	Ожидаемый Результат	Статус
N-1	TR-1	Запуск		1. Нажать Пуск 2. Выбрать Notepad	Окно Notepad с пустым файлом открыто	Passed
N-2	TR-2	Создать новый файл		1. Меню Файл 2. Выбрать Создать	1. Создан новый файл 2. Рабочая область пустая	Passed
N-3	TR-3	Ввод текста	1. Файл Notepad открыт	1. Набрать несколько слов 2. Удалить несколько слов	1. В рабочей области приложения появлются (отображаются) слова 2. Слова пропадают (стираются)	Failed
N-4	TR-4	Сохранение файла	1. Файл Notepad открыт 2. Текст введен	1. Меню Файл 2. Выбрать Сохранить 3. выбрать каталог для сохранения файла 4. Ввести имя файла 5. Нажать кнопку Сохранить	1. Диалоговое окно для сохранения файла закрыто 2. На диске в указанном каталоге появился файл с заданным именем	Blocked
N-5	TR-5	Распечатать файл	1. Файл Notepad открыт 2. Текст введен	1. Меню Файл 2. Распечатать 3. Задать параметры печати: - принтер - количество копий - вид печати 4. Подтвердить печать	1. Диалоговое окно Распечатать закрыто 2. Документ с выбранными параметрами распечатан успешно	Blocked

Задача о треугольнике




STORMNET